Report: Movie Search Application

For the development of this Movie Search app, I started with a base structure from a YouTube tutorial. The tutorial provided a great starting point for working with the OMDB API to fetch movie data and display search results. I built upon this base to introduce additional features like movie rating and collection functionality, and improved the user experience with a few enhancements.

What I Learned from the Tutorial:

1. **Fetching Data from OMDB API**: The tutorial showed how to send requests to the OMDB API using the fetch() method. By using the API's endpoint with a search term, I learned how to retrieve basic movie details like title, year of release, and poster image. The tutorial demonstrated how to handle the API response by parsing the JSON and dynamically populating the movie list.
2. **Displaying Search Results**: I learned to dynamically create HTML elements for each movie search result by iterating through the API response. It involved creating div elements for each movie, storing the IMDb ID in the data-id attribute, and inserting the movie title and poster image into the corresponding HTML structure.
3. **Displaying Movie Details**: The tutorial also covered displaying detailed movie information. Once a user clicked on a movie, a detailed API request would fetch additional information such as genre, director, actors, plot, and awards, and show it in a dedicated area of the page.

What I Built Upon:

While the tutorial provided a solid foundation, I added several features and improvements to make the app more functional and engaging.

1. **Improved UI/UX:**

   I managed the visibility of the search results dynamically. For example, the movie search list is hidden automatically if the input box is empty or when a movie is clicked. I utilized the classList.add() and classList.remove() methods to handle these visibility changes smoothly, giving a more fluid user experience.
2. **Movie Collection:**

I added functionality to allow users to store their favorite movies in localStorage. The stored movies persist after the page is refreshed, ensuring the collection remains even if the user leaves the site. Each movie is saved with its title, poster, and IMDb ID. Users can also remove movies from their collection by clicking a "remove" button.

3. **Rating System:**

   A key feature I implemented is the rating system. I wanted users to rate movies they've watched. To do this, I added star elements for each movie in the detailed view. The star elements are div elements styled as empty stars by default. When a user hovers over a star, it fills with color, indicating the rating they're about to give.

   To handle this, I used event listeners for mouse hover (mouseenter) and mouse click (click). On hover, the stars fill accordingly, and once clicked, the rating is stored in localStorage for the corresponding movie.

   I also implemented a system to display the current rating when users revisit the movie's details. This involved checking localStorage for existing ratings and updating the star elements based on the stored value.

4. **Event Management:**

   I refined event listeners and user interaction throughout the app. For instance, clicking outside the search input box or search results list hides the list to ensure that the UI remains clean and uncluttered. This creates a more intuitive experience as users interact with the app.

Issues Encountered:

The problem I faced for the longest time was that while the API worked fine locally (on localhost), the movie content didn't display correctly when deployed on GitHub Pages. After, embarrassingly, over an hour of debugging, I found the issue: I had mistakenly used http instead of https in the API request URL, causing a security conflict. Correcting this trivial error resolved the issue, and the deployed app now works as intended.

Git and Version Control:

Throughout the development process, I worked with Git and GitHub to manage the project's versions and track changes. I've become more comfortable with commands like git add, git commit, git push, and git pull. Additionally, I learned how to handle merging branches and

resolving conflicts. Using Git for version control made it easier to manage updates, especially as I iterated on features.

Additional Features in Progress

Trending Now Feature

I wanted to add a "Trending Now" feature to showcase popular movies across genres. I created the frontend for this feature, but I encountered a limitation with the OMDB API: it does not provide ratings or popularity data to determine which movies are trending. To work around this, I implemented a random function that imports six random movies from each genre. To simplify navigation, I grouped multiple tags from the API as a single genre.

Future Enhancements

I have started working on a future version of the app, where I aim to integrate a recommendation system. This system will use a machine learning model leveraging natural language processing (NLP) to group together keywords and tags from movie descriptions. The goal is to recommend movies similar to the ones users rate highly and avoid suggesting those rated poorly. This feature will provide a more personalized movie discovery experience.